Manipal Institute of Technology

Aetherion

A Procedural 3D Raycasting Horror Engine

Submitted by:

Siddharth	240961220
Tarkshya	240961242
Jogith	240961070

Academic Year: 2025 – 2026

Abstract

Aetherion is a minimal, SDL2-based raycasting engine designed to evoke the ambience of early 3D horror games such as DOOM and Silent Hill. The project explores low-level game loop design, texture rendering, and real-time event handling using the C++ SDL2 library.

1 Introduction

Modern 3D engines abstract much of the complexity of rendering pipelines. Aetherion reconstructs this process by implementing core systems audio, rendering, and input-using SDL2.

2 Objectives

- To design a lightweight procedural rendering engine using C++ and SDL2.
- To implement real-time audio and rendering subsystems.
- To explore event-driven architecture for interactive graphics.

3 Software and Hardware Requirements

Software

- C++17 compiler (g++ or Clang)
- SDL2, SDL2_image, and SDL2_mixer libraries
- Linux / Windows / macOS

Hardware

- Processor: Dual-core or higher
- RAM: 4 GB minimum
- Display: 1080p recommended

4 Getting Started

Prerequisites

Ensure the following are installed:

- C++ Compiler (GCC 9+ or Clang 10+)
- CMake 3.22+
- SDL2 and SDL2_image libraries

Installing Dependencies on Linux

```
# Ubuntu/Debian
sudo apt-get install build-essential cmake libsdl2-dev libsdl2-image-dev

# Fedora
sudo dnf install gcc-c++ cmake SDL2-devel SDL2_image-devel

# Arch Linux
sudo pacman -S base-devel cmake sdl2 sdl2_image
```

Installing on NixOS

```
# Enter the Nix shell environment
nix-shell

# Or install system-wide
nix-env -iA nixos.cmake nixos.SDL2 nixos.SDL2_image nixos.gcc
```

Installing on macOS and Windows

```
# macOS (using Homebrew)
brew install cmake sdl2 sdl2_image

# Windows
# Download libraries from official sources:
# SDL2: https://github.com/libsdl-org/SDL/releases
# SDL2_image: https://github.com/libsdl-org/SDL_image/releases
```

Building the Project

```
git clone https://github.com/mooofin/HEKATE.git
cd Aetherion
mkdir build && cd build
cmake ..
make
```

Running the Game

```
./doom
```

Project Architecture

```
HEKATE/
          src/
                main.cpp          # Game loop
                raycaster.h       # Raycasting logic
                imageloader.h     # Texture management
                color.h           # Color definitions
          assets/
                audios/
                maps/
                screens/
                textures/
          build/
          CMakeLists.txt
          README.md
```

Table 1: Game Controls

Key	Action
W / ↑	Move forward
S / ↓	Move backward
A	Strafe left
D	Strafe right
←/ →	Rotate camera
F	Toggle flashlight
ESC	Exit game

5 Methodology

Block Diagram

Important Code Snippets

Main Rendering Loop:

Listing 1: Main rendering loop handling events and drawing.

```
1 while (running) {
2     SDL_Event event;
```

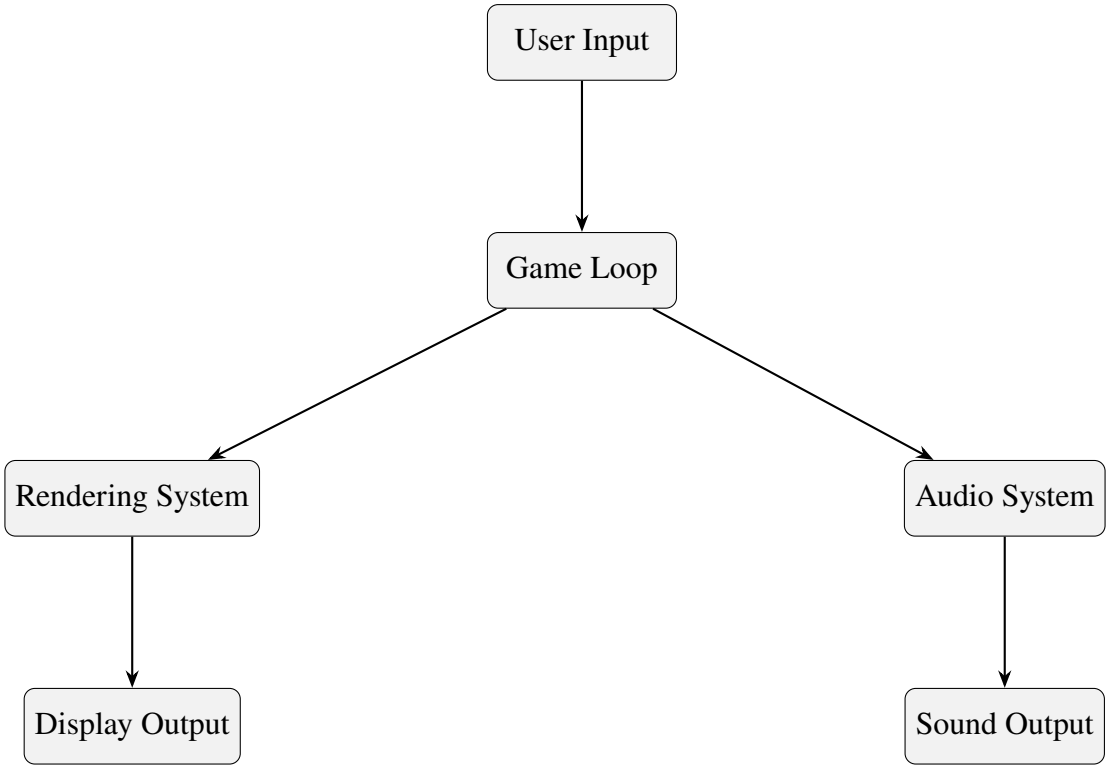

Figure 1: Overall Block Diagram of Aetherion Architecture

```
    while (SDL_PollEvent(&event)) {
        if (event.type == SDL_QUIT) running = false;
        if (event.type == SDL_KEYDOWN) {
            switch(event.key.keysym.sym) {
                case SDLK_UP:    r.moveForward(); break;
                case SDLK_DOWN:  r.moveBackward(); break;
                case SDLK_LEFT:  r.rotateLeft();   break;
                case SDLK_RIGHT: r.rotateRight();  break;
                case SDLK_SPACE: toggleFlashlight(); break;
            }
        }
    }
    clear();
    r.render();
    draw_ui(player);
    SDL_RenderPresent(renderer);
}
```

Audio Handling:

Listing 2: Loading and playing a .wav file with SDL2.

```
int PlayMusic() {
    SDL_AudioSpec wavSpec;
    Uint32 wavLength;
    Uint8* wavBuffer;
```

```
5
6       if (SDL_LoadWAV( gameAnthem.wav , &wavSpec, &wavBuffer, &wavLength) ==
            nullptr)
7           return 1;
8
9       SDL_OpenAudio(&wavSpec, nullptr);
10      SDL_QueueAudio(1, wavBuffer, wavLength);
11      SDL_PauseAudio(0);
12
13      SDL_Delay(wavLength * 1000 / wavSpec.freq);
14
15      SDL_CloseAudio();
16      SDL_FreeWAV(wavBuffer);
17      return 0;
18  }
```

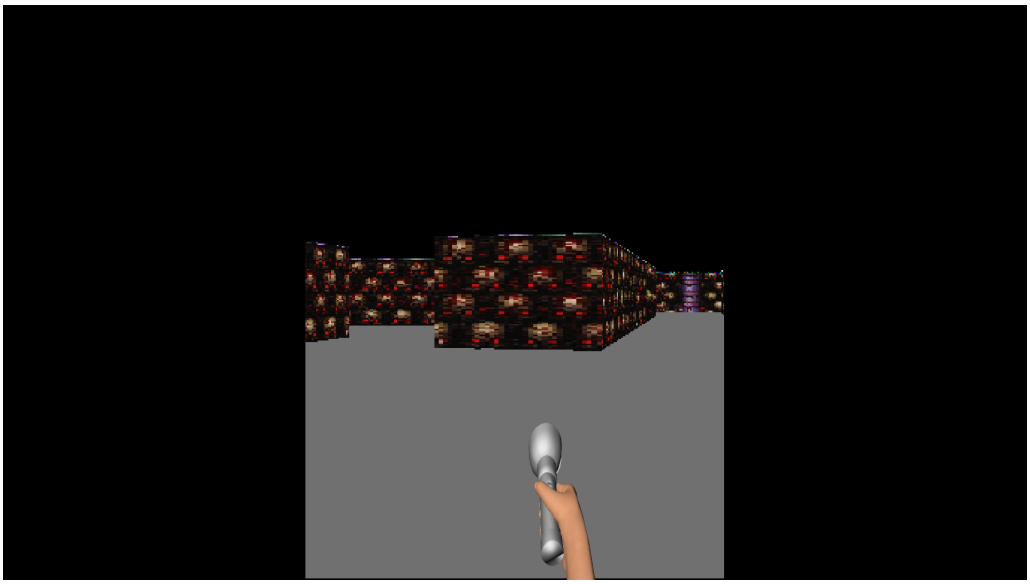

Figure 2: Gameplay output of the Aetherion project

6 Conclusion

Aetherion demonstrates the feasibility of building a functional raycasting engine using minimal dependencies. It provides insights into graphics programming, resource management, and real-time system design. Future work includes advanced lighting, AI, and procedural generation.

References

[1] SDL2 Documentation. https://wiki.libsdl.org/

[2] Fabien Sanglard. *Doom Source Code Analysis.*

[3] Lazy Foo' Productions. *SDL2 Tutorials.* https://lazyfoo.net/tutorials/SDL/